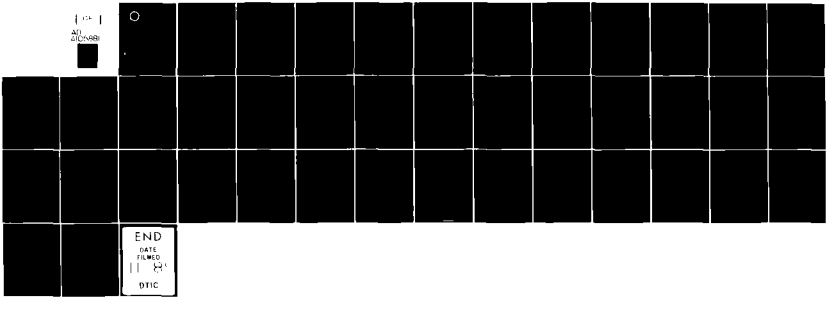
AD-A105 881 STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB F/G 12/1
EXACT AND APPROXIMATION ALGORITHMS FOR A SCHEDULING PROBLEM,(U)
JUL 81 G DOBSON N00014-75-C-0267
UNCLASSIFIED SOL-81-9 NL



END
DATE FILMED
DTIC

Systems
Optimization
Laboratory

LEVEL II

1

AD A105881

DTIC FILE COPY

Department of Operations Research
Stanford University
Stanford, CA 94305

81 10 14

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

EXACT AND APPROXIMATION ALGORITHMS FOR A SCHEDULING PROBLEM

by

Gregory Dobson

TECHNICAL REPORT SOL 81-9

July 1981

Research and reproduction of this report were partially supported by the Department of Energy Contract AM03-76SF00326, PA# DE-AT03-76ER72018; Office of Naval Research Contract N00014-75-C-0267; National Science Foundation Grants MCS-7681259, MCS-7926009 and ECS-8012974.

Exact and Approximation Algorithms for a Scheduling Problem

Gregory Dobson
Systems Optimization Laboratory
Department of Operations Research
Stanford University
Stanford, California 94305

Abstract

This paper discusses problems that arose in calendaring cases for an appellate court. The first problem is to distribute cases among panels of judges so as to equalize work loads. We give a worst case analysis of a heuristic for this $\mathcal{NP}$-complete problem. For a given distribution denote by z the heaviest work load. We wish to minimize z. The ratio of the heuristic value $\bar{z}$ to that of the true optimum z^* is shown to be $\bar{z}/z^* \leq (k+3)/(k+2)$ where all the case weights are in $[0,(1/k)z^*]$, generalizing a result of Graham on multiprocessor scheduling. Under a restrictive assumption on the case weights, some generalizations of this scheduling problem are solved. Characterizations for feasible calendars and polynomial algorithms for finding these feasible solutions are given. Algorithms are given for choosing an optimal subset of the backlogged cases that can be calendared.

1. Problem Statement

The following problem arose from calendaring cases for the U. S. 9th Circuit Court of Appeals. We are given a large collection of cases each with a weight assessing its difficulty (and thus its requirement for court time), and the date it arrived in the queue of the backlogged cases (used as an indicator of its priority). Each month there are, say, b panels of judges each with the same capacity to hear cases. Normally the judges are all circuit (higher) court judges who may hear any case. Occasionally it is necessary, however, for the court to borrow judges from the district (lower) courts. A judge from district x is not allowed to hear a case from district x. At most one district judge is assigned to a panel. Given these assignments, the problem is to find an optimal, in some sense, subset of the backlogged cases and assign them to panels so that the panels receive equal work loads and no district judge hears a case from his/her district.

We abstract the problem to the following coloring problem. There is a large collection of items (cases), I. For each item j we associate a weight, w_j, and a cost (arrival date), t_j. There is a set of bins (panels), B. Bin k has a capacity s_k. Furthermore we assign a 'color' to each district and give each item (case) j the color c_j of its district. If a panel has a district x judge then the associated bin k is colored $d_k \equiv x$. The restriction is that an item colored x cannot be placed in a bin colored x. Our problem is to find the optimal collection $J \subseteq I$ that solves

$$\begin{aligned} &\underset{J \subseteq I}{\text{maximize}}\ t(J) \\ &\quad \text{s. t. } P = (P_1, \ldots, P_b) \text{ being a partition of } J \\ &\qquad w(P_i) \text{ are 'equal' and 'within capacity'} \\ &\qquad c_j \neq d_k \text{ for } j \in P_k, k \in B. \end{aligned} \tag{Q_0}$$

Here $t(J) = \sum_{j \in J} t_j$ and $w(P_k) = \sum_{j \in P_k} w_j$.

Bin Packing can be seen to be a special case of Q_0 if we ignore the color restriction and the problem of equalizing the amount per bin and if we set $t(j) = 1$ and interpret '$w(P_i)$ within capacity' to be $w(P_i) \leq 1$. 3-Partition can be viewed as a special case of Bin Packing so both Bin Packing and 3-Partition appear as subproblems; each is $\mathcal{NP}$-complete in the strong sense (Garey and Johnson 1979). In view of this it is interesting to know for packing problems of the above sort what kind of heuristics will work well and under what restrictive assumptions these problems can be solved in polynomial time. In that the weights of items are rough estimates of the court time required, it is reasonable to see if some advantage can be gained by either interpreting the weight values liberally, or treating the constraint $w(P_k)$ 'equal' and 'within capacity' liberally.

In §2 we consider just the problem of distributing cases among panels without district restrictions (items among bins without color restrictions) so as to have 'nearly equal' work loads. In the remainder of this paper the item weights are assumed to be in a restricted class. With this assumption we are able to look at more complicated problems. In §3 we discuss the questions of feasibility and optimality for problem (Q_0) when the color restriction is ignored. In §4 deals with the same questions for problem (Q_0) with the color restriction, but here it is necessary to add a further condition that we have one 'color free' bin. In §5 we drop this 'color free' bin assumption and characterize feasible packings with color restrictions.

2. Packing with Arbitrary Weights

The main objective in the calendaring problem is to schedule cases to provide 'full' and 'equal' work loads for the panels. We formulate the problem of

packing items, I, with arbitrary weights into bins, B, as evenly as possible as

$$z^* \equiv \underset{P}{\text{minimize}} \ \underset{1 \le i \le b}{\text{maximum}} \ w(P_i)$$

where the minimization is over all partitions $P = (P_1, \ldots, P_b)$ of I. $b \equiv |B|$. This problem is $\mathcal{NP}$-hard since it contains the $\mathcal{NP}$-complete problem, 3-Partition. To see this observe that if the 3-Partition problem has bins of size K then there exists a way to place the items into the b bins of size K if there is a partition of the items that gives a value of $z^* \le K$. This section considers the heuristic: Place the largest unpacked item in the bin that has the least amount in it so far; repeat until all items are packed. We will call this LIME for 'Largest In Most Empty'. Let $\bar{P} = (\bar{P}_1, \ldots, \bar{P}_b)$ be a partition of I given by LIME, and denote $\max_{1 \le i \le b} w(\bar{P}_i)$ by $\bar{z}$. This heuristic is also called LPT (Longest Processing Time) heuristic in the literature. The problem is identical to that of multiprocessor scheduling of independent tasks to minimize the latest finishing time. One would suspect that the worst-case error, $\bar{z}/z^*$, would be small if all the items were relatively small.

Theorem 2.1. If for all $j \in I$, $w_j \in [0, \frac{1}{k} z^*]$ and if the LIME heuristic gives a partition $\bar{P}$ with value $\bar{z}$ then

$$\frac{\bar{z}}{z^*} \le \frac{k+3}{k+2}$$

and the bound is tight.

Note that $w_j \le z^*$ so that $k \ge 1$ always holds and the ratio $\bar{z}/z^* \le 4/3$. This was shown by Graham (1969) under the guise of multiprocessor scheduling. To apply the theorem to obtain a better than 4/3 bound requires prior knowledge of z^*. Note though, that $(k'+3)/(k'+2)$ over estimates this ratio when $k' = \lfloor (w(I)/b) \max_{j \in I} (1/w_j) \rfloor$ since $w(I)/b$ is a lower bound on z^*.

Scaling. We may assume without loss of generality that $z^* = 1$. By scaling each weight by $1/z^*$ (i.e. $w_j \leftarrow w_j/z^*$) we have that the new scaled value of $z^* = 1$ and that $0 \leq w_j \leq 1$ for all $j \in I$.

We begin the proof of Theorem 2.1 with

Lemma 2.2. If $w_j \leq \alpha$ for all $j \in I$ then for the LIME heuristic

$$\frac{\bar{z}}{z^*} < 1 + \alpha.$$

Proof. Let $s = w(I)/b$. Claim: if bin i is filled to a level of at least s then we never place another item in bin i. Assume not. Some item j is placed in bin i with a level of at least s. Every other bin is filled to a level of at least s since i was the most empty bin. Hence $w(I) > w(\{1, \ldots, j-1\}) \geq bs$, contradicting the definition of s. Clearly $s \leq z^*$ and if j is placed in a bin of level $t < s$ then $t + w_j < s + \alpha \leq z^* + \alpha \equiv 1 + \alpha$.∎

Proof of 2.1. Assume that there is a first item j that overfills bin i to a level of at least $\frac{k+3}{k+2}$. First observe that $w_j \notin [0, \frac{1}{k+2}]$ by Lemma 2.2. Second, if $w_j \in (\frac{1}{k+1}, \frac{1}{k}]$ and it overfills a bin to a level $\frac{k+3}{k+2}$, then the level in every bin must be at least

$$\frac{k+3}{k+2} - \frac{1}{k} = \frac{(k+3)k - (k+2)}{k(k+2)} = \frac{k^2 + 2k - 2}{k(k+2)}$$
$$= 1 - \frac{2}{k(k+2)} > \frac{k-1}{k}$$

Thus every bin must have k items in it. Certainly no bin can hold more than k such items and have a level under z^*. Hence there is no way to pack items $\{1, \ldots, j\}$ so that the maximum level is at most z^*, contradicting the definition of z^*.

It remains to show that $w_j \notin (\frac{1}{k+2}, \frac{1}{k+1}]$.

Case $k = 1$. Here we assume j is the first item to exceed level z^*. If $w_j \in (\frac{1}{3}, \frac{1}{2}]$ then there are either one or two items in each bin (after packing $\{1, \ldots, j-1\}$). Say the heuristic places item j in bin i. If there is only one item, k, in the bin i, so $w_j + w_k > 1$ then clearly $w_k + w_l > 1$ for every $l \in \{1, \ldots, j\}$. Hence any optimal packing of $\{1, \ldots, j\}$ must have item k in a bin by itself. Since all other items stored one to a bin are larger than k, they too must be by themselves in any optimal packing of $\{1, \ldots, j\}$. Every other bin has 2 items in it. If j is to be placed without exceeding z^* level then 3 items must go in some bin but $w_k > \frac{1}{3}$ for all $k \leq j$, contradiction. ∎

Case $k = 2$. Item j is the first to exceed level $\frac{5}{4}$. If $w_j \in (\frac{1}{4}, \frac{1}{3}]$ then we can write $w_j = \frac{1}{3} - \epsilon$ where $0 \leq \epsilon < \frac{1}{12}$. Each bin is filled to a level at least $\frac{5}{4} - \frac{1}{3} + \epsilon = \frac{11}{12} + \epsilon$. There can be either 2, 3 or 4 items in a bin. If there are 2 then the larger one weighs at most $\frac{1}{2}$ so the smaller, r, weighs at least $\frac{5}{12} + \epsilon$. Item r can only be packed in a bin with 2 items in an optimal packing since if we compute the total weight of item r and the 2 two smallest items we have

$$\begin{aligned} w_r + w_{j-1} + w_j &\geq \frac{5}{12} + \epsilon + 2\left(\frac{1}{3} - \epsilon\right) \\ &= \frac{13}{12} - \epsilon > 1. \end{aligned}$$

Here every item that is packed in a bin with 2 items can only fit in a bin with 2 items in an optimal packing of $\{1, \ldots, j\}$. Every bin can only take 3 so there is at least one too many items. ∎

Case $k \geq 3$. We assume that $w_j \in (\frac{1}{k+2}, \frac{1}{k+1}]$ and that j is the first item to exceed a level of at least $\frac{k+3}{k+2}$. We will show that in fact the items $\{1, \ldots, j\}$ could not be packed in b bins of size 1.

Claim 1: Let $w_j = \frac{1}{k+1} - \epsilon$ where $0 \leq \epsilon < \frac{1}{(k+2)(k+1)}$. The level in the most empty bin is at least

$$1 - \frac{1}{(k+1)(k+2)} + \epsilon$$

Proof.

$$\begin{aligned}\frac{k+3}{k+2} - \frac{1}{k+1} + \epsilon &= \frac{(k+3)(k+1) - (k+2)}{(k+1)(k+2)} + \epsilon \\ &= \frac{k^2 + 3k + 2 \quad -1}{(k+1)(k+2)} + \epsilon \\ &= 1 - \frac{1}{(k+1)(k+2)} + \epsilon\end{aligned}$$

∎

Claim 2: Every bin has k, $k+1$ or $k+2$ items in it.

Proof. If it had $k+3$ items then since each item weighs strictly more than $\frac{1}{k+2}$, the current level would be strictly greater than $\frac{k+3}{k+2}$, contradicting the definition of j being the first item to exceed that level. If it had $k-1$ items then the current level would be at most

$$\frac{k-1}{k} < 1 - \frac{1}{(k+1)(k+2)} + \epsilon$$

contradicting claim 1.∎

Claim 3: There is at least one bin, r, with k elements.

Proof. If this were not the case then every bin has at least $k+1$ items and there would be at least $(k+1)b+1$ items all of weight greater than $\frac{1}{k+2}$, but only $k+1$ of these items could possibly fit in a bin of capacity $z^* = 1$. This contradicts the definition of z^*.∎

Claim 4: If $x_1 \geq \ldots \geq x_k$ are weights of items in bin r, the one with k items, then

$$x_k > \frac{1}{k+1} + \epsilon$$

Proof.

$$\begin{aligned} x_k &\geq 1 - \frac{1}{(k+1)(k+2)} + \epsilon - x_1 - \cdots - x_{k-1} \\ &\geq 1 - \frac{1}{(k+1)(k+2)} + \epsilon - \frac{k-1}{k} \\ &= \frac{(k+1)(k+2) - k}{k(k+1)(k+2)} + \epsilon \\ &= \frac{k^2 + 2k + 2}{k(k+1)(k+2)} + \epsilon \\ &> \frac{1}{k+1} + \epsilon \end{aligned}$$

∎

Claim 5: Let $y_1 \geq \cdots \geq y_{k+1}$ be the weights of items in a bin q containing $k+1$ items, then

$$y_{k-1} > \frac{1}{k+1} + \epsilon$$

Proof. If not then by claim 4 and the fact that the items are packed by decreasing weight, the $k-1$st item was placed after the kth item in the r bin (the one with k items). Thus the level in the q bin is at least

$$
\begin{aligned}
& x_1 + \cdots + x_{k-1} + y_{k-1} + y_k + y_{k+1} \\
\geq & 1 - \frac{1}{(k+1)(k+2)} + \epsilon - \frac{1}{k} + 3\left(\frac{1}{k+1} - \epsilon\right) \\
= & 1 + \frac{3}{k+1} - \frac{1}{k} - \frac{1}{(k+1)(k+2)} - 2\epsilon \\
> & 1 + \frac{3k - (k+1)}{k(k+1)} - \frac{3}{(k+1)(k+2)} \\
= & 1 + \frac{(2k-1)(k+2) - 3k}{k(k+1)(k+2)} \\
= & 1 + \left(\frac{1}{k+2}\right)\left(\frac{2k^2-2}{k(k+1)}\right) \\
\geq & \frac{k+3}{k+2}
\end{aligned}
$$

since

$$\frac{2k^2-2}{k(k+1)} \geq 1$$

iff $2k^2 - 2 - k(k+1) \geq 0$

iff $k^2 - k - 2 \geq 0$

iff $k(k-1) - 2 \geq 0$

iff $k \geq 2$

contradicting that j was the first to exceed the level $\frac{k+3}{k+2}$. ∎

It is clear that if bin q had had $k+2$ items of weights $y_1 \geq \cdots \geq y_{k+2}$ then

$$y_{k-1} > \frac{1}{k+1} + \epsilon$$

Claim 6: Let r be the number of bins with k items. Let s be the number of bins with at least $k+1$ items. Then we have by the previous claims that there are at least $kr + (k-1)s$ 'large' items which weigh strictly more than $\frac{1}{k+1} + \epsilon$, and at least $2s+1$ 'small' items which weigh at least $\frac{1}{k+1} - \epsilon$. The

claim is that any repacking of the bins so that some bin q has $k+1$ items and a level of at most $z^* = 1$ must have at least 3 small items.

Proof. Assume you can do it with only 2. The level in such a bin with $k-1$ large and 2 small items is strictly greater than

$$\begin{aligned}&(k-1)\left(\frac{1}{k+1}+\epsilon\right)+2\left(\frac{1}{k+1}-\epsilon\right)\\ =&1+(k-3)\epsilon \geq 1\end{aligned}$$

provided $k \geq 3$. ∎

To finish the proof of the upper bound in Theorem 5.1 observe that at least $s+1$ bins must have $k+1$ items. The only way to place $k+1$ items into bins of capacity 1 is to have 3 small items. We can create at most $\lfloor \frac{2s+1}{3} \rfloor$ bins with $k+1$ items by claim 6. This contradiction of the definition of z^* shows that $w_j \notin (\frac{1}{k+2}, \frac{1}{k+1}]$.

To see that the bound is tight consider the following examples parameterized by $k = 1, 2, 3, \ldots$. We will produce an example that has an error of at least $\frac{k+3}{k+2} - \delta$. Let n be large enough so that $\delta \equiv \frac{1}{n}\left(\frac{1}{k+1} - \frac{1}{k+2}\right)$ is 'small' enough. We defined δ so that $\frac{1}{k+2} + n\delta = \frac{1}{k+1}$. The list of items is

$$\begin{array}{lll}
k+1 & \text{of weight} & \frac{1}{k+2}+(2n-1)\delta \\
\vdots & \vdots & \vdots \\
k+1 & \text{of weight} & \frac{1}{k+2}+(n+1)\delta \\
k+1 & \text{of weight} & \frac{1}{k+2}+n\delta \\
(k-1)(k+1) & \text{of weight} & \frac{1}{k+2}+n\delta = \frac{1}{k+1} \\
k+1 & \text{of weight} & \frac{1}{k+2}+(n-1)\delta \\
\vdots & \vdots & \vdots \\
k+1 & \text{of weight} & \frac{1}{k+2}+1\delta
\end{array}$$

$k+2$ of weight $\frac{1}{k+2}+0\delta$

There are $(k+1)n$ bins. The LIME heuristic places the items in the bins so that there are $k+1$ identical sets of n bins as displayed below. Here $\alpha=\frac{1}{k+2}$ and each bin contains $k+1$ items. The $k-1$ middle ones are all of weight $\frac{1}{k+2}+n\delta=\frac{1}{k+1}$.

(a)	(b)		(y)	(z)
$\alpha+0\delta$	$\alpha+1\delta$	$\cdots$	$\alpha+(n-2)\delta$	$\alpha+(n-1)\delta$
$\alpha+n\delta$	$\alpha+n\delta$	$\cdots$	$\alpha+n\delta$	$\alpha+n\delta$
$\vdots$ $\vdots$	$\vdots$ $\vdots$		$\vdots$ $\vdots$	$\vdots$ $\vdots$
$\alpha+n\delta$	$\alpha+n\delta$	$\cdots$	$\alpha+n\delta$	$\alpha+n\delta$
$\alpha+(2n-1)\delta$	$\alpha+(2n-2)\delta$	$\cdots$	$\alpha+(n+1)\delta$	$\alpha+n\delta$

n bins repeated $k+1$ times

The $k+2$nd item of weight $\frac{1}{k+2}$ is not shown. Observe that each bin is filled to a level $1-\delta$. To see that all the items can be stored in $(k+1)n$ bins of capacity 1 we move the item of weight $\frac{1}{k+2}+0\delta$ out of bin (a) and place it aside. Next move the item which weighs $\frac{1}{k+2}+1\delta$ out of bin (b) and into bin (a). Move the item which weighs $\frac{1}{k+2}+2\delta$ out of bin (c) and into bin (b) and so on. At this point all the bins labeled (a),(b), ...,(y) are filled to level 1, there are $k+2$ items of weight $\frac{1}{k+2}$ on the side and all the (z) bins have k items of weight $\frac{1}{k+2}+n\delta=\frac{1}{k+1}$. It is now possible to place all the items of

weight $\frac{1}{k+2}$ in one bin and all the remaining items of weight $\frac{1}{k+1}$ (there are $k(k+1)$ of them) in the remaining k bins. The new packing looks like

(a)	(b)		(x)	(y)
$\alpha + 1\delta$	$\alpha + 2\delta$	$\cdots$	$\alpha + (n-2)\delta$	$\alpha + (n-1)\delta$
$\alpha + n\delta$	$\alpha + n\delta$	$\cdots$	$\alpha + n\delta$	$\alpha + n\delta$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$\alpha + n\delta$	$\alpha + n\delta$	$\cdots$	$\alpha + n\delta$	$\alpha + n\delta$
$\alpha + (2n-1)\delta$	$\alpha + (2n-2)\delta$	$\cdots$	$\alpha + (n+2)\delta$	$\alpha + (n+1)\delta$

$n-1$ bins repeated $k+1$ times

And the remaining bins labeled (z) look like

(z)	(z)		(z)	(z)
				$\alpha + 0\delta$
$\alpha + n\delta$	$\alpha + n\delta$	$\cdots$	$\alpha + n\delta$	$\alpha + 0\delta$
$\alpha + n\delta$	$\alpha + n\delta$	$\cdots$	$\alpha + n\delta$	$\alpha + 0\delta$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$\alpha + n\delta$	$\alpha + n\delta$	$\cdots$	$\alpha + n\delta$	$\alpha + 0\delta$
$\alpha + n\delta$	$\alpha + n\delta$	$\cdots$	$\alpha + n\delta$	$\alpha + 0\delta$

3. Packing with Nested Weights

We return now to the optimization problem introduced at the beginning. Given a set of items, I, and a set of bins, B, find the optimal collection $J \subseteq I$ that solves

$$\begin{aligned} &\underset{J \subseteq I}{\text{maximize}}\ t(J) \\ &\quad \text{s.t. } P = (P_1, \ldots, P_b) \text{ being a partition of } J \\ &\qquad w(P_i) \text{ are 'equal' and 'within capacity'} \\ &\qquad c_j \neq d_k \text{ for } j \in P_k, k \in B. \end{aligned} \tag{Q_0}$$

In this chapter, in order to make this problem more tractable we assume that the weights are restricted to be in a set $V = \{v_1, \ldots, v_n\}$ where $v_i/v_{i+1} \in Z$, e.g. $\{1, 1/3, 1/6, 1/12, \ldots\}$. If the weights of I satisfy this assumption we call them *nested*.

Notation.

C is the set of colors

I is the set of items

I^x is the set of items colored x

I_i is the set of items of weight v_i

I_i^x is the set of items colored x weighing v_i

B is the set of bins ($b \equiv |B|$)

B^x is the set of bins colored x

V is the set of possible weight values ($n \equiv |V|$)

An item colored x we refer to as an x-item. We also call an item weighing v_i, a v_i-item. Which is meant will be clear from context.

We start with the question of feasibility, i.e. when does there exist a packing, that is a partition $P = (P_1, \ldots, P_b)$ of I so that $w(P_k) = s_k$ for

$k = 1, \ldots, b$ where s_k is the capacity of bin $k \in B$. The following is almost immediate from our assumption about the weights being nested.

Theorem 3.1. There exists a packing of items, I, into bins, B, if and only if

$$w\Big(I_1 \bigcup \cdots \bigcup I_i\Big) \le \sum_{k \in B} \lfloor s_k \rfloor_i \quad \text{for } i = 1, \ldots, |V| \tag{3.1}$$

where $\lfloor y \rfloor_i = v_i \lfloor y/v_i \rfloor$. We can find a packing in $O(|I| + |V|)$ time.

Proof. The proof is by induction on $|I|$. If $|I| = 1$ and $I = \{j\}$ where $w_j = v_{\hat{i}}$ then

$$v_{\hat{i}} = w\Big(I_1 \bigcup \cdots \bigcup I_{\hat{i}}\Big) \le \sum_{k \in B} \lfloor s_k \rfloor_{\hat{i}}$$

so that some $s_{\hat{k}} \ge v_{\hat{i}}$. Place item j in bin $\hat{k}$. Now assume we have a problem with $|I| = q$. Let j be an item of largest weight in I, $w_j = v_{\hat{i}}$.

$$w(I_{\hat{i}}) = w\Big(I_1 \bigcup \cdots \bigcup I_{\hat{i}}\Big) \le \sum_{k \in B} \lfloor s_k \rfloor_{\hat{i}}$$

so there is some bin $\hat{k}$ with $s_{\hat{k}} \ge v_{\hat{i}}$. Place item j in bin $\hat{k}$. Remove item j from I calling it $\bar{I}$ and reduce $s_{\hat{k}}$ by $v_{\hat{i}}$. So $\bar{s}_{\hat{k}} = s_{\hat{k}} - v_{\hat{i}}$, and $\bar{s}_k = s_k$ for $k \ne \hat{k}$. Clearly (3.1) still holds for $i = 1, \ldots, \hat{i} - 1$. For $i \ge \hat{i}$

$$\begin{aligned} w\Big(\bar{I}_1 \bigcup \cdots \bigcup \bar{I}_i\Big) &= w\Big(I_1 \bigcup \cdots \bigcup I_i\Big) - v_{\hat{i}} \\ &\le \sum_{\substack{k \in B \\ k \ne \hat{k}}} \lfloor s_k \rfloor_i + \lfloor s_{\hat{k}} \rfloor_i - v_{\hat{i}} \\ &= \sum_{\substack{k \in B \\ k \ne \hat{k}}} \lfloor \bar{s}_k \rfloor_i + v_i \lfloor s_{\hat{k}}/v_i \rfloor - v_i (v_{\hat{i}}/v_i) \end{aligned}$$

since $v_{\hat{i}}/v_i$ is an integer

$$\begin{aligned} &= \sum_{\substack{k \in B \\ k \ne \hat{k}}} \lfloor \bar{s}_k \rfloor_i + v_i \left\lfloor \frac{s_{\hat{k}} - v_{\hat{i}}}{v_i} \right\rfloor \\ &= \sum_{k \in B} \lfloor \bar{s}_k \rfloor_i \end{aligned}$$

Thus (3.1) holds for $\bar{I}$ and $(\bar{s}_k)_{k\in B}$, and by induction there exists a solution.

The induction suggests the following $O(|I| + |V|)$ algorithm to find the packing. Create $|V|$ buckets for items and sort the items by weight using a bucket sort. Do the same for the bins placing bin k in bucket v_i if $s_k \geq v_i$ but $s_k < v_{i-1}$. Here it is assumed that the weights w_j and the capacities s_k are such that the identification $w_j = v_i$ or $s_k \geq v_i, s_k < v_{i-1}$ can be made in $O(1)$ time. Remove an item of largest weight, remove a bin from the bucket with largest label and place the item in the bin, thus reducing the capacity of the bin. Return the bin to the appropriate bucket. The sorts take $O(\max(|I|, |V|))$ time. The remainder requires $O(|I|)$ time. If the algorithm attempts to pack an item of weight v_i into a bin of capacity s_k with $s_k < v_i$ then (3.1) is violated. ∎

If in fact we wish to fill each bin to capacity, $w(P_k) = s_k$, then it is only necessary to change the last inequality in (3.1).

Theorem 3.2. There exists a packing $P = (P_1, \ldots, P_b)$ of I into bins B such that $w(P_k) = s_k$ if and only if

$$w\Big(I_1 \bigcup \cdots \bigcup I_i\Big) \leq \sum_{k\in B} \lfloor s_k \rfloor_i \quad \text{for } i = 1, \ldots, |V| - 1$$
$$w(I) = \sum_{k\in B} s_k$$

Using Theorem 3.2 we are able to solve the optimization problem which is formally

$$\begin{aligned} &\underset{J\subseteq I}{\text{maximize}}\ t(J) \\ &\qquad P = (P_1, \ldots, P_b) \text{ partitions } J \\ &\qquad w(P_k) = s_k \end{aligned} \tag{Q_3}$$

Theorem 3.3. (Q_3) can be solved in $O(|I| \log |I|)$ time.

Proof. Using Theorem 3.2 we have that

$$\begin{aligned} w(I_i \bigcup \cdots \bigcup I_n) &= w(I) - w\Big(I_1 \bigcup \cdots \bigcup I_{i-1}\Big) \\ &\geq \sum_{k \in B} s_k - \sum_{k \in B} \lfloor s_k \rfloor_{i-1} \\ &= \sum_{k \in B} (s_k - \lfloor s_k \rfloor_{i-1}) \end{aligned}$$

Since every feasible solution fills every bin there must be at least

$$\sum_{k \in B} (s_k - \lfloor s_k \rfloor_{n-1})/v_n \text{ items of weight } v_n$$

to fill the top segments of these bins where $s_k - \lfloor s_k \rfloor_{n-1} > 0$. For example if $V = \{1, \frac{1}{3}\}$ and $s_k = 1\frac{2}{3}$ then we need at least $\frac{2}{3}/\frac{1}{3} = 2$ items of weight $\frac{1}{3}$ to fill this bin. If there are any additional items of weight v_n then they will come in integral multiples of v_{n-1}/v_n. The procedure is to sort the items of weight v_n by decreasing t_j values and choose the best $\sum_{k \in B}(s_k - \lfloor s_k \rfloor_{n-1})/v_n$ of them. Next, group the remaining v_n-items into packets of cardinality v_{n-1}/v_n. Call these packets G_l, $l = 1, \ldots, q$. Pack the best v_{n-1}/v_n in G_1, the next best in G_2 and so forth. The justification for this procedure is that if there are going to be

$$r \equiv \sum_{k \in B} (s_k - \lfloor s_k \rfloor_{n-1})/v_n + m\frac{v_{n-1}}{v_n} \text{ for } m = 1, 2, \ldots$$

in the solution then they will clearly be the r with largest t_j's. Place the $\sum_{k \in B}(s_k - \lfloor s_k \rfloor_{n-1})/v_n$ items in the bins where $s_k - \lfloor s_k \rfloor_{n-1} > 0$ and reduce their capacities. Replace each packet, G_l, and the associated v_n-items by one v_{n-1}-item with $\bar{t}_{G_l} \equiv \sum_{j \in G_l} t_j$. The problem has been reduced to one with no v_n-items. Formally the induction argument is: if $|V| = |\{v_1\}| = 1$ the optimal subset J^* is the one with $\sum_{k \in B} s_k/v_1$ best items. If we have a

problem with $|V| = n$ apply the procedure above to reduce the problem to one where $|V| = n-1$. By induction we find a subset J^* that is optimal to the smaller problem. By the argument above J^* with the $\sum_{k\in B}(s_k - \lfloor s_k \rfloor_{n-1})/v_n$ items already chosen is optimal to the original problem.

The procedure to find the optimal subset J^* is

$J^* \leftarrow \emptyset$

$s_k \leftarrow \lfloor s_k \rfloor_n$ for $k = 1 \ldots, b$

for $i \leftarrow n, \ldots, 1$ do begin
  sort I_i by profit values
  remove $\sum_{k\in B}(s_k - \lfloor s_k \rfloor_{i-1})/v_i$ best items from I_i
    and place them in J^*
  $s_k \leftarrow \lfloor s_k \rfloor_{i-1}$ for $k \in B$
  while $I_i \neq \emptyset$ do begin
    remove best v_{i-1}/v_i items from I_i, call it E
    add one v_{n-1}-item to I_{i-1} with profit $t_E = \sum_{j\in E} t_j$
    end
  end

It remains to justify the time bound. There are $|I_i|$ items of weight v_i in I. Once the sorting is completed at any particular stage it requires only linear time to group the items, so clearly the sorting is dominate in the time bound. Let $n = |V|$ then there are n sorts to be done. The first stage sorts $|I_n|$ items, the second $|I_{n-1}| + \lfloor |I_n| v_n / v_{n-1} \rfloor$, and the third

$$\begin{aligned} &|I_{n-2}| + \left\lfloor \frac{v_{n-1}}{v_{n-2}} \left(|I_{n-1}| + \left\lfloor \frac{v_n}{v_{n-1}} |I_n| \right\rfloor \right) \right\rfloor \\ \leq &|I_{n-2}| + \frac{v_{n-1}}{v_{n-2}} |I_{n-1}| + \frac{v_n}{v_{n-2}} |I_n|. \end{aligned}$$

the kth stage sorts at most

$$\sum_{j=0}^{k-1} \frac{v_{n-j}}{v_{n-(k-1)}} |I_{n-j}|$$
$$\leq \sum_{j=0}^{k-1} \left(\frac{1}{2}\right)^j |I_{n-j}|.$$

If T is the total time for sorting,

$$T = O\Big(\sum_{k=1}^{n}\Big(\sum_{j=0}^{k-1}\Big(\frac{1}{2}\Big)^j |I_{n-j}|\Big)\log\Big(\sum_{j=0}^{k-1}\Big(\frac{1}{2}\Big)^j |I_{n-j}|\Big)\Big)$$
$$\leq O\Big(\log|I|\Big(\sum_{j=0}^{n-1}\Big(\frac{1}{2}\Big)^j \sum_{k=j+1}^{n} |I_{n-j}|\Big)\Big)$$
$$\leq O\Big(\log|I|\Big(\sum_{j=0}^{n-1}\Big(\frac{1}{2}\Big)^j\Big)|I|\Big)$$
$$\leq O(|I|\log|I|)$$

∎

A slight modification of the above procedure will allow it to handle the problem where each bin is not required to be completely full. The problem is formally

$$\begin{aligned} &\underset{J\subseteq I}{\text{maximize}}\ t(J) \\ &\qquad P = (P_1, \ldots, P_b) \text{ partitions } J \\ &\qquad w(P_k) \leq s_k \end{aligned} \tag{Q_4}$$

First if $t_j < 0$ then we may drop item j. Next we may consider each set I_i of items, to be padded with enough zero profit items ($t_j = 0$) so that every item is either placed in J^* or is passed up as a part of a larger item into I_{i-1}. Call this new set of items $\bar{I}$. We now solve (Q_3) with the augmented set $\bar{I}$. To justify this consider the augmented problem with a large number of items of weight v_n and $t_j = 0$. We solve

$$\begin{aligned} &\underset{\bar{J}\subseteq \bar{I}}{\text{maximize}}\ t(\bar{J}) \\ &\qquad \bar{P} = (\bar{P}_1, \ldots, \bar{P}_b) \text{ partitions } \bar{J} \\ &\qquad w(\bar{P}_k) = s_k \end{aligned} \tag{Q_5}$$

If $\bar{P} = (\bar{P}_1, \ldots, \bar{P}_b)$ is a partition of $\bar{J}$ that solves (Q_5), then let $J = \bar{J} \bigcap I$ and $P_k = \bar{P}_k \bigcap I$ and we claim that $P = (P_1, \ldots, P_b)$ which is a partition of J is optimal to (Q_4). Let $R = (R_1, \ldots, R_b)$ be a partition of K an optimal solution to (Q_4). Hence $t(K) \geq t(J)$. Add $(s_k - w(R_k))/v_n$ dummy items of weight v_n and profit 0 to R_k and thus K, creating $\bar{K}$. This is feasible to (Q_5). We have

$$t(K) = t(\bar{K}) \leq t(\bar{J}) = t(J).$$

Hence J is optimal to (Q_4). ∎

Corollary 3.4. If each bin k, with capacity s_k is such that $s_k/v_1 \in Z$ then there exists a packing of I into bins B if and only if $w(I) \leq \sum_{k \in B} s_k$.

Proof. If $s_k/v_1 \in Z$ then $s_k = \lfloor s_k \rfloor_i$ for all i, so the nth inequality implies the rest and

$$w(I) = w\Big(I_1 \bigcup \cdots \bigcup I_n\Big) \leq \sum_{k \in B} \lfloor s_k \rfloor_n = \sum_{k \in B} s_k. \qquad \blacksquare$$

Example. Let us solve a problem of the form (Q_0) with $|B| = 2$, $s_1 = 1$, $s_2 = 1\frac{1}{3}$, $V = \{1, \frac{1}{3}, \frac{1}{6}\}$, $|I| = 10$, where $\{(w_j, t_j) \mid j \in I\} = \{(1,7), (1,8), (\frac{1}{3},3), (\frac{1}{3},2), (\frac{1}{3},2), (\frac{1}{3},4), (\frac{1}{6},1), (\frac{1}{6},2), (\frac{1}{6},1), (\frac{1}{6},3)\}$. First we sort the items of size $\frac{1}{6}$ by 't' values to get: $(\frac{1}{6},3)$, $(\frac{1}{6},2)$, $(\frac{1}{6},2)$, $(\frac{1}{6},1)$. We then combine them into items of size $\frac{1}{3}$: $(\frac{1}{3},5)$, $(\frac{1}{3},3)$. Now sort all the items of size $\frac{1}{3}$: $(\frac{1}{3},5)$, $(\frac{1}{3},4)$, $(\frac{1}{3},3)$, $(\frac{1}{3},3)$, $(\frac{1}{3},2)$, $(\frac{1}{3},2)$. Remove the best one for the space of size $\frac{1}{3}$ in bin 2, and then combine the rest into items of size 1: $(1,10)$. Sort all the items of size 1: $(1,10)$, $(1,8)$, $(1,7)$. Pick the best 2 for the spaces of size 1 in bins 1 and 2. The optimal solution is: $(\frac{1}{3},4)$, $(\frac{1}{3},3)$, $(\frac{1}{6},2)$, $(\frac{1}{6},1)$ in bin 1 and $(1,8)$, $(\frac{1}{6},3)$, $(\frac{1}{6},2)$ in bin 2.

Note. If $s_k/v_1 \in Z$ for all k the optimization problem (Q_4) is just the knapsack problem

$$\underset{J \subseteq I}{\text{maximize}}\ t(J)$$
$$w(J) \leq \sum_{k \in B} s_k$$

where the weights, w_j, are nested. Although this problem is NP-hard for arbitrary weights by Theorem 3.3 it is solvable in $O(|I| \log |I|)$ time.

4. Packing with the Color Restriction

We now ask when will a given set of colored items, I, exactly pack into b colored bins, B, all of capacity s,$(s/v_1 \in Z)$ and meet the color restriction. We assume without loss of generality that $s = 1$.

Denote by I^x the items colored x i.e. $\{j \mid j \in I, c_j = x\}$. A simple necessary condition is

$$w(I^x) \leq b - |B^x| \quad \text{for all } x \in C \tag{4.1}$$

To see this observe that if in fact $w(I^x) > b - |B^x|$ then the total weight of items colored x exceeds the available space in bins not colored x. If we assume further that all items have the same weight then (1) is sufficient. This is just a special case of Hall's theorem for perfect matchings but it will be useful in § 5.

Theorem 4.1. If all items in I have the same weight then there is a packing into the bins B that satisfies the color restriction if and only if

$$w(I) = b \text{ and } w(I^x) \leq b - |B^x| \text{ for all } x \in C. \tag{4.2}$$

Further, we can find a maximum packing in $O(|I|)$ time.

We give the algorithm first because this special case can be solved faster than the general case of matching, $O(|I|)$ rather than $0(|I|^{2.5})$ time, see Even and Kariv (1975), and second to introduce some ideas we use later.

Proof of 4.1. The necessity of the condition was shown above. To see sufficiency assume (4.2) holds and apply the algorithm: Place items into any available bins into which they are allowed. Assume this procedure stops at some point without packing all the items. Space can only be available in bins of one color, say x, and the only remaining unpacked items must also be colored x. Search the non-x-bins for one that contains a non-x-item. If you find one replace the item with an x-item and repack this replaced item in some x-bin. Thus we can assume that every non-x-bin is filled with x-items. If there are still more x-items to be packed then clearly $w(I^x) > b - |B^x|$ contradicting (4.2). It remains to show that this algorithm can be implemented in $0(|I|)$ time.

First we may assume without loss of generality that the bins have been broken down into 'subbins' all of capacity v_1. Assume that the colors are labeled $1, \ldots, |C|$. Create $|C|$ buckets for bins. Place the bins into buckets according to color. Create $|C|$ buckets for items and sort the items by color. This requires $O(|B| + |I| + |C|)$ time and we assume here that $|B| \leq |I|$ and $|C| \leq |I|$. Next we perform the following:

```
i ← 1
b ← 2
while ( there remain items to be packed ) do begin
    if bin-bucket b is empty then b ← b + 1
    else if item-bucket i is empty then begin
        i ← i + 1
        if i = b then begin
            place any bins in bucket b into the bottom of bucket 1
            b ← b + 1
            end
        end
    else begin
        remove item from bucket i
        remove bin from bucket b
        place item in bin
        end
    end
```

Clearly the algorithm executes the loop in $O(|I| + |C|)$ time since each time we either pack an item or we pass over a bin or item-bucket. If an item colored x is placed in a bin colored x then the number of items colored x is greater than

$$|B^{x+1}| + \cdots + \left|B^{|C|}\right| + |B^1| + \cdots + |B^{x-1}|$$

thus we have

$$w(I^x) > b - |B^x|$$

contradicting our assumption. ∎

Obviously for nested weights the necessary conditions (4.2) are not sufficient. As an example consider a 2 bin problem, one red and one blue bin,

with 3 items, one red item of weight $\frac{1}{2}$, one blue item of weight $\frac{1}{2}$ and a white item of weight 1. The red item must go in the blue bin, the blue item must go in the red bin, and there is no space for the large white item. It is somewhat surprising then that if there is just one 'color free' bin, i.e. one that will accept any item, then (4.2) is sufficient.

Theorem 4.2. If B contains one 'color free' bin, and J is a set of items with nested weights then there exists a packing of J into bins, B, that meets the color restriction if and only if

$$w(J) = b \text{ and } w(J^x) \leq b - |B^x| \text{ for all } x \in C. \tag{4.3}$$

Before we prove Theorem 4.2 we need a lemma.

Lemma 4.3. Given a set of items J, $w(J) = 1$ and let j be an item of smallest weight, then for all $v \in V$, $v \geq w_j$, there exists a subset K that includes item j such that $w(K) = v$.

Proof. Let $J_{\text{large}} = \{k \in J \mid w_k \geq v\}$ $J_{\text{small}} = J - J_{\text{large}}$. The capacity consumed by the small items, J_{small}, is a multiple of v. Create $w(J_{\text{small}})/v$ subbins of capacity v. Sort the items in J_{small} in decreasing order by weight, i.e. $w_1 \geq \cdots \geq w_j$, so that the jth item is last. Pack J_{small} into the subbins via First Fit Decreasing (Johnson et al. 1974), i.e. place items in the order specified above in the first available subbins that have sufficient capacity. We can do this by Corollary 3.4. Clearly the contents of the last subbin is the desired set K. K contains j and $w(K) = v$.∎

Proof of 4.2. The proof of sufficiency is again by an algorithm that finds the packing. First sort the items into decreasing order by weight. Place items into any bins into which they will fit until you are stuck. Because the items

are being packed in decreasing order any remaining capacity in a bin is a multiple of the weight of the item currently being packed. In fact, if not all the items are packed then we are currently trying to place an item j and the only available capacity is in bins colored c_j.

Temporarily place the item j, in some c_j-bin.

Case 1: The 'color free' bin is not completely filled with c_j-items:
In this case there exists an item k, of weight $w_k \geq w_j$ not colored c_j in the 'color free' bin. If $w_k = w_j$ exchange items j and k. If $w_k > w_j$ first fill the bin containing item j with dummy items of weight w_j. Apply the lemma to obtain a subbin of items K, $j \in K$ and $w(K) = w_k$. Exchange the subbin K with item k and remove the dummy items from the bins.

Case 2: Every item in the 'color free' bin is colored c_j:
Search all the packed non-c_j-bins for one, say i, containing a non-c_j-item, k. This must exist by (4.3). Otherwise every non-c_j-bin would be filled with c_j-items and if we include the item j then

$$w(I^{c_j}) > b - |B^{c_j}|$$

violating (4.3). Exchange the complete contents of the 'color free' bin with that of bin i found above. Notice that this does not violate the color restriction since bin i is not colored c_j and we are only placing c_j-items in it and any item legally goes into the 'color free' bin. Further, the non-c_j-item, k, is now in the 'color free' bin so we are back in case 1.∎

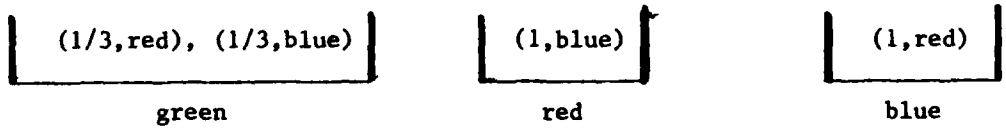
Corollary 4.4. If the necessary conditions (4.3) hold then there exists a packing of items, J, into bins, B, that violates the color restriction in at most one bin.

Proof. Temporarily remove the color from any bin, i. Since there is now a 'color free' bin and (4.3) holds we apply Theorem 4.2 to obtain a packing

that meets the color restriction. Now repaint bin i with its original color. Clearly this is the only bin that could violate the color restriction. ∎

Example. Consider the problem with $|B| = 3$ with one green, one blue and one red bin each of size 1. Let $V = \{1, \frac{1}{3}\}$ and $\{(w_j, c_j) \mid j \in I\} = \{(1, red), (1, \text{blue}), (\frac{1}{3}, \text{blue}), (\frac{1}{3}, \text{red}), (\frac{1}{3}, \text{yellow})\}$. First observe that there are no green items so the green bin can act as a color free bin. We pack the items in decreasing order: (1, red) into the green bin, (1, blue) into the red bin, $(\frac{1}{3}, \text{red})$ into the blue bin. At this point we are blocked from legally placing the next item, $(\frac{1}{3}, \text{blue})$ into the blue bin as shown below.

Since the color-free bin is not completely filled with blue items we exchange the smallest non-blue item in the color free bin with something in the blue bin to get



The other item, $(\frac{1}{3}, \text{yellow})$, may now be placed.

Note. The existence of a color free bin is not necessary. Consider the 2 bin example: one red bin and one blue bin, 4 items of weight $\frac{1}{2}$ one red, one blue, and two green.

If we continue to assume that B contains a 'color free' bin then the optimization problem (Q_0) reduces to

$$\begin{aligned} \underset{J \subseteq I}{\text{maximize}}\ & t(J) \\ & w(J) = b \\ & w(J^x) \leq b - |B^x| \text{ for all } x \in C \end{aligned} \tag{Q_6}$$

where J^x is the set of items colored x in J. This can be solved via a dynamic programming recursion.

Theorem 4.5. (Q_6) can be solved by a dynamic programming recursion in $O(|C|(|B|/v_n)^2 + |I|(|B|/v_n))$ time where v_n is the smallest weight in V.

Proof. Let $I^E \equiv \bigcup_{x \in E} I^x$ and define for a subset $E \subseteq C$,

$$\begin{aligned} f^E(s) = \underset{J \subseteq I^E}{\text{maximum}}\ & t(J) \\ & w(J) = s \\ & w(J^x) \leq b - |B^x| \text{ for all } x \in E. \end{aligned}$$

For a given subset of colors E, $f^E(s)$ is the value of the best set of items with colors in E, which has total weight s and meets the necessary conditions (4.3). In particular

$$\begin{aligned} f^{\{x\}}(s) = \underset{J \subseteq I^x}{\text{maximum}}\ & t(J) \\ & w(J) = s \\ & w(J^x) \leq b - |B^x| \end{aligned} \tag{4.4}$$

and $f^{\{x\}}(s)$ is defined to be $-\infty$ for $s > b - |B^x|$. First compute $f^{\{x\}}(s)$ for all colors $x \in C$ and all s. Then recursively compute $f^C(t)$.

$$f^{E \bigcup \{x\}}(t) = \underset{0 \leq s \leq t}{\text{maximum}} \{f^{\{x\}}(s) + f^E(t-s)\} \tag{4.5}$$

The validity of the above recursion is seen by observing that the optimal solution to the problem (Q_6) restricted to colors $E \bigcup \{x\}$ with $b = t$ must

contain some amount of x-items from 0 to t. We simply maximize over all such combinations. It should be clear that the subset J^* giving value $f^C(b)$ is the solution to our problem. The computation of $f^{\{x\}}(s)$ for all s requires $O((|B|/v_n)|I^x|)$ time via the standard dynamic programming recursion for the knapsack problem. To compute $f^{\{x\}}(s)$ for all $x \in C$ requires $O((|B|/v_n)\sum_{x\in C}|I^x|) = O((|B|/v_n)|I|)$ time. Each application of the recursion (4.5) requires $O(|B|/v_n)$ time. To compute this for all values of $t \in \{0, v_n, 2v_n, \ldots, b\}$ requires $O((|B|/v_n)^2)$ time and we need to compute this $|C|$ times to find $f^C(b)$. Hence the total time bound is

$$O((|B|/v_n)|I| + (|B|/v_n)^2|C|).$$

∎

5. Characterizations for Packings with Color Restriction

We now characterize packing problems that have solutions in terms of the existence of flows in a particular kind of network.

For the moment we return to the case $|V| = 1$ and by Theorem 3.1 the conditions $w(I^x) \leq b - |B^x|$ for all $x \in C$ are sufficient. If we turn this inequality around and think of the $|B^x|$ as variable for a second we have that

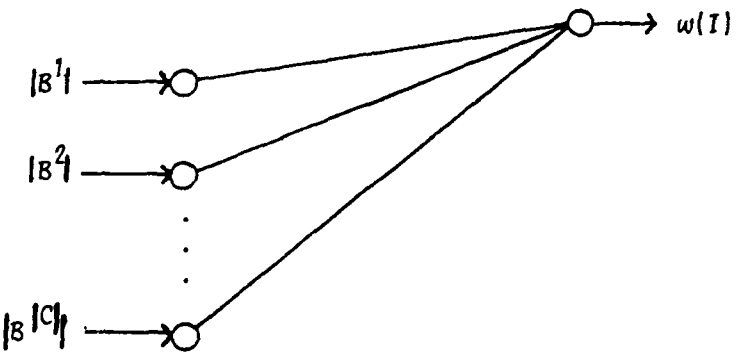
$$|B^x| \leq b - w(I^x) = w(I) - w(I^x) \text{ for all } x \in C$$

or we can pack all the items (not necessarily filling the bins) if and only if

$$\sum_{x\in C} \max(|B^x|, w(I) - w(I^x)) \geq w(I).$$

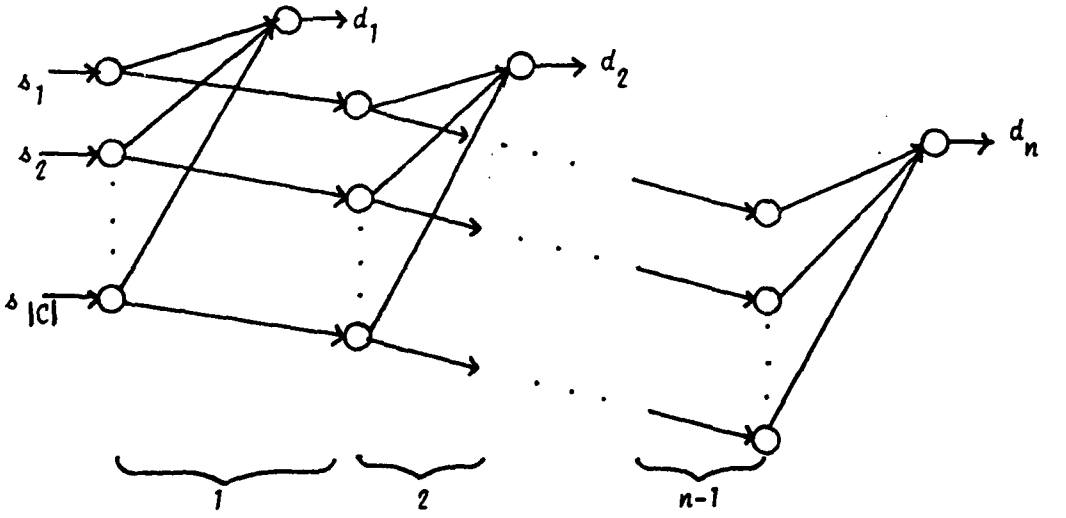
Looking at this from a network point of view let there be a supply node with supply $|B^x|$ for each $x \in C$, one demand node with a demand of $w(I)$, and a

capacity of $w(I) - w(I^x)$ on the arc from source x to the sink. We have that there exists a packing of I into B if and only if there exists a flow of value at least $w(I)$ in the network. This is pictured below.



We now jump to the general case.

Definition 5.1. A *cascade* is a network of the following form: there are $|C|$ sources. Source $x \in C$ has a supply of $s_x \equiv |B^x|$. There are $|V|$ sinks. Sink $v_i \in V$ has a demand of $d_i \equiv v_i|I_i|$, i.e. total weight of items with weight v_i. The structure is shown below.



The cascade is divided into $|V|$ sections. The flow in the arcs of section i is restricted to be an integral multiple of v_i. The capacities on the arcs in section i, denoted by c_{xi} and d_{xi} for the 'down' and 'up' arcs respectively that are connected to source x, are given by

$$\begin{aligned} d_{xi} &= d_i - w(I_i^x) \quad \text{for } i = n, \ldots, 1 \\ c_{xn} &= 0 \\ c_{xi-1} &= \lfloor (d_{xi} + c_{xi}) \rfloor_i \quad \text{for } i = n, \ldots, 0 \end{aligned} \tag{5.1}$$

Definition 5.2. We say that there is a *feasible flow* in the cascade if there exists a flow (f, g), where f_{xi}, (g_{xi}) are the flows in the down (up) arcs, such that

(capacity)	$f_{xi} \le c_{xi}$	$g_{xi} \le d_{xi}$	(5.2a)
(conservation of flow)	$f_{xi-1} = f_{xi} + g_{xi}$		(5.2b)
(supply = demand)	$f_{x0} = s_x$	$\sum_{x \in C} g_{xi} = d_i$	(5.2c)
	$f_{xi}/v_i \in Z$	$g_{xi}/v_i \in Z$	(5.2d)

In words a flow that uses the supply, meets the demand, satisfies conservation of flow and further the flow in the arcs of section i of the cascade are an integral multiple of v_i.

The importance of the cascade to the packing problem is

Theorem 5.3. Every packing induces a feasible flow and every feasible flow induces a packing.

Proof. Assume we have a packing $\{P^x\}_{x \in C}$ where P^x is the set of items that are packed into bins colored x. We now apply the following algorithm

to compute f and g

$$
\begin{aligned}
&f = g = 0 \\
&\text{for } x \in C \text{ do begin} \\
&\quad \text{for } i = 1, \ldots, |V| \text{ do} \\
&\quad\quad \text{for each item } j \in I_i \bigcap P^x \text{ do} \\
&\quad\quad\quad g_{xi} \leftarrow g_{xi} + v_i \\
&\quad \text{for } i = |V|, \ldots, 1 \text{ do} \\
&\quad\quad f_{xi-1} \doteq f_{xi} + g_{xi} \\
&\quad \text{end.}
\end{aligned}
$$

We now claim that (f, g) satisfy (5.2) and thus constitutes a feasible flow. Clearly (f, g) satisfies (5.2b) conservation of flow since this the way it was computed. As for (5.2a) we have

$$
\begin{aligned}
g_{xi} &= w(\{ \text{ all items of weight } v_i \text{ in bins colored } x\}) \\
&\leq w(\{ \text{ all items of weight } v_i \text{ not colored } x\}) \\
&= d_{xi}
\end{aligned}
$$

By a simple induction on i we have that

$$
\begin{aligned}
f_{xi-1} &= \lfloor (f_{xi-1}) \rfloor_i \\
&= \lfloor (f_{xi} + g_{xi}) \rfloor_i \\
&\leq \lfloor (c_{xi} + d_{xi}) \rfloor_i \\
&\equiv c_{xi-1}
\end{aligned}
$$

To see (5.2c) observe that

$$
\begin{aligned}
\sum_{x \in C} g_{xi} &= \sum_{x \in C} w(\{\text{all items of weight } v_i \text{ in bins colored } x\}) \\
&= w(\{\text{all items of weight } v_i\}) \\
&= d_i
\end{aligned}
$$

$$
\begin{aligned}
f_{x0} &= \sum_{1 \leq i \leq |V|} w(\{\text{all items of weight } v_i \text{ in bins colored } x\}) \\
&= |B^x| \equiv s_x
\end{aligned}
$$

Finally for (5.2d) we know,

$$g_{xi}/v_i \in Z$$

since it was computed as a sum of v_i's. To see $f_{xi}/v_i \in Z$, observe that all the bins are full (thus all x-bins are full). We thus have that items of weight v_n come in groups of cardinality v_{n-1}/v_n, thus f_{xn-1} is a multiple of v_{n-1}. Now viewing each group of v_{n-1}/v_n items of weight v_n as 1 item of weight v_{n-1} we may proceed inductively to see that f_{xi} is a multiple of v_i.

Now assume we have a feasible flow in the cascade we wish to show how this induces a packing. Let (f, g) be a flow, and for each $x \in C$ divide the x-bins into smaller subbins so that there are g_{xi}/v_i subbins of capacity v_i. This can be done since $\sum_{1 \le i \le |V|} g_{xi} = s_x$. Now for each weight $v_i \in V$ we need to place the items of weight v_i. We know this can be done by Theorem 3.1 if and only if

$$v_i(\text{number of bins of capacity } v_i \text{ and color } x) + w(I_i^x) \le d_i$$

if and only if

$$g_{xi} \le d_i - w(I_i^x) = d_{xi}$$

But this is just the capacity constraint on the flow. Thus there is a way to pack the items into the subbins as they have been divided above. ∎

The second characterization is actually a corollary of Theorem 4.2. The idea is simply if the packing problem can be divided into two packing problems, the first one having a color free bin and the second has a solution then there is a solution to the entire problem and conversely.

Theorem 5.4. Let J be a set of items and B be a set of bins then there is a feasible packing of J into B if and only if there exists for some color x

a partition of $J = J_1 \bigcup J_2$ and $B = B_1 \bigcup B_2$ such that $J_1 \bigcap J^x = \emptyset$ and $B_1 \bigcap B^x \neq \emptyset$ and there is a feasible packing of J_2 into B_2.

Proof. ($\Leftarrow$) This half is obvious from Theorem 4.2 since the bin in $B_1 \bigcap B^x$ acts as a 'color free' bin and J_2 pack into B_2 by assumption.

($\Rightarrow$) Pick a color x for which there is an item j. Take a feasible packing and let B_1 be the set containing the bin with item j. Let J_1 be the set of items in this bin.∎

Unfortunately neither of these characterizations appears to lead to a polynomial algorithm. There are no known techniques for solving flows in cascade networks. In fact it is not hard to see that a cascade can be converted to a network with capacities and gains on the nodes. Thus it is closely related to a $\mathcal{NP}$-complete problem. Here the gains would be 1 and v_{k-1}/v_k for $v_k \in V$. Thus for $V = \{1, \frac{1}{2}, \frac{1}{4}, \dots\}$ the only gains would be 1 and 2.

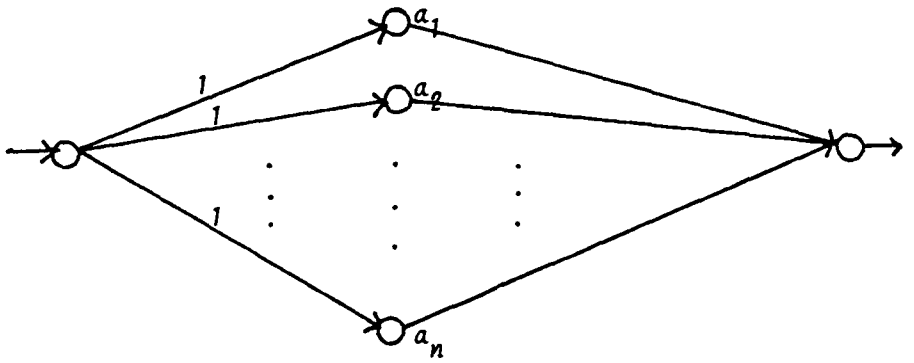
Theorem 5.5. Given a network with multipliers on the nodes and capacities on the arcs, the question: is there a integral flow of at least K, is $\mathcal{NP}$-complete even if all gains are either 1 or 2.

Proof. Let $\{\{a_j\}_{j=1}^n, b\}$ be an instance of Partition. The question: is there a subset $S \subseteq \{1, \dots, n\}$ such that $\sum_{j \in S} a_j = b/2$ is $\mathcal{NP}$-complete. We show how to convert this instance to a network with capacities on arcs and gains of 1 and 2 on the nodes, so that the desired subset corresponds to a integral flow of $b/2$.

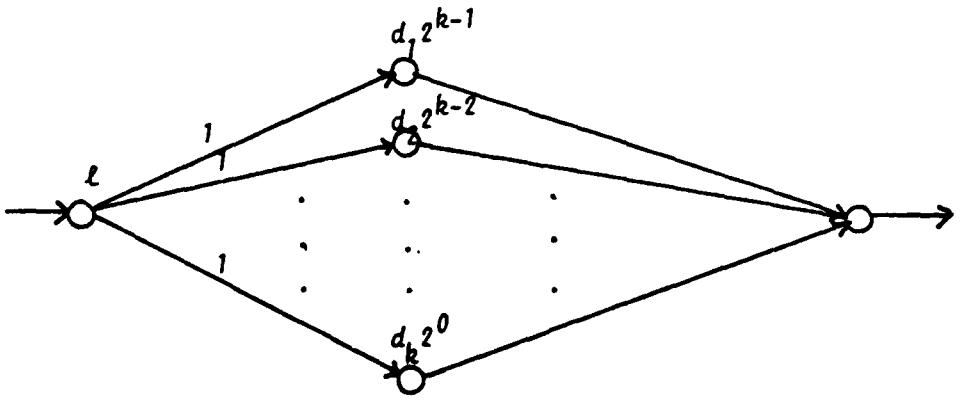
First consider the network

of 1 and 2 on the nodes, so that the desired subset corresponds to a integral flow of $b/2$.

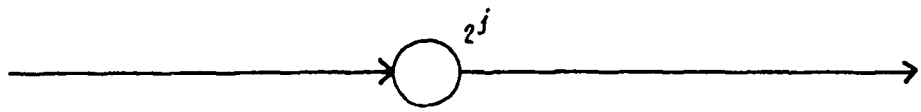
First consider the network



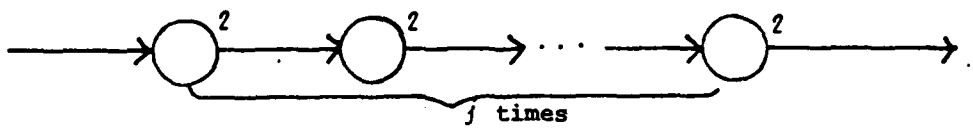
where the capacities are the number on the arcs and the gains are the numbers on the nodes. Clearly Partition has a solution if and only if this network has an integral flow of $b/2$. We now show how to reduce each node with a gain of a to a network with gains of only 1 and 2, using $O(\log^3 a)$ nodes. Let the binary expansion of a be $d_1 d_2 \ldots d_k$ where $d_i = 0, 1$ and $l = \sum_{1 \le i \le k} d_i$. consider the network



We now convert each



to



We now have $O(k^2)$ nodes. We now repeat the process for the node with multiplier of l. The number of ones in the binary expansion of l is less than the number of ones in the binary expansion of a, so we need only repeat this process at most $\lceil \log a \rceil$ times. We require $O(\log^3 a)$ nodes in the section of the network representing the original node with gain a. We have thus reduced Partition to an instance of a network with arc capacities and nodes with gains of 1 and 2. The network has at most

$$O\Big(\sum_{1 \le j \le n} \log^3 a_j\Big) \text{ nodes.}$$

The reduction is polynomial and the network problem is $\mathcal{NP}$-complete.∎

For a final result we return to the problem of simply distributing the weights evenly among the bins.

Theorem 4.4. If I is a set of nested weights then the LIME heuristic gives an optimal solution to

$$z^* \equiv \underset{P}{\text{minimize}} \ \underset{1 \le i \le b}{\text{maximum}} \ w(P_i)$$

where the minimization ranges over all partitions $P = (P_1, \ldots, P_b)$ of I.

Proof. By induction on $|I|$. Clearly it is true for $|I| = 1$. Let $|I| = k$, let j be one of the items of least weight and apply the heuristic to $\hat{I} = I - \{j\}$. By induction the heuristic gives an optimal partition $\hat{P} = (\hat{P}_1, \ldots, \hat{P}_b)$ of $\hat{I}$. Denote $\max_{1 \le i \le b} w(\hat{P}_i)$ by s. If the most empty bin, q, has $w(\hat{P}_q) < s$ then by the fact that the weights are nested, $s - w(\hat{P}_q)$ is an integral multiple of w_j. Place item j in bin q and the value of the solution does not increase. Hence it is an optimal partition of I. Otherwise $w(\hat{P}_l) = s$ for all $l = 1, \ldots, b$, so we may place item j in any bin. Since w_j is the smallest weight, the optimal value must be an integral multiple of w_j. A lower bound for z^* is $w(I)/b > s$ but the value of the solution found is the smallest multiple of w_j greater than s and thus is optimal.∎

References

Aho, A., Hopcroft, J., and Ullman, J., (1974). *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading, Mass.

Coffman, E. G. Jr., (ed.) (1976). *Computer and Job-Shop Scheduling Theory*, John Wiley and Sons, New York.

Coffman, E. G., Jr., Garey, M., and Johnson, D., (1978). "An Application of Bin-Packing to Multiprocessor Scheduling" *SIAM J. Comput.* **7**, 1–17.

Cook, S. A., (1971). "The Complexity of Theorem-Proving Procedures," *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 151–158. **17**, 449–467.

Even, S. and Kariv, O. (1975). "An $O(n^{2.5})$ Algorithm for Maximum Matching in General Graphs," *Proc. IEEE Symp. Foundations of Computer Sci.*, **16**, 100–112.

Garey, M., Graham, R., and Johnson, D., (1978). "Performance Guarantees for Scheduling Algorithms," *Opns. Res.* **26**, 3–21.

Garey, M. R., Graham, R. L., Johnson, D. S. and Yao, A. C. (1976). "Resource Constraint Scheduling as Generalized Bin Packing," *J. Combinatorial Theory Ser. A.*, **21**, 257–298.

Garey, M. R., and Johnson, D. S., (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freedman and Co., San Francisco.

Garey, M., Johnson, D., (1976). "Approximation Algorithms for Combinatorial Problems: An Annotated Bibliography," in *Algorithms and Com-*

plexity: Recent Results and New Directions, 41–52, Traub, J., (ed.), Academic Press.

Gonzalez, T., Ibarra, O., and Sahni, S., (1977). "Bounds for LPT Schedules on Uniform Processors," *SIAM J. Comput.* **6**, 155–166.

Graham, R. L. (1976). "Bounds on the Performance of Scheduling Algorithms," in *Computer and Job/Shop Scheduling Theory*, Coffman, E. G., ed. John Wiley, New York, chapter 5.

Graham, R. L. (1969). "Bounds on Multiprocessing Timing Anomalies," *SIAM Journal on Applied Mathematics*, **17**, 2, 416–425.

Graham, R. L., (1966). "Bounds on Certain Multiprocessing Timing Anomalies," *SIAM J. Appl. Math.* **45**, 1563–1581.

Gonzalez, T., and Sahni, S., "Flowshop and Jobshop Schedules: Complexity and Approximation," *Opns. Res.* **26**, 36–52.

Ibarra, O., and Kim, C., (1977). "Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors," *J. Assoc. Comput. Mach.* **24**, 280–289.

Johnson, D., Demers, A., Ullman, J., Garey, M., and Graham, R., (1974). "Performance Bounds for Simple One-Dimensional Bin Packing Algorithms," *SIAM J. Comput.* **3**, 299–325.

Karp, R., (1972). "Reducibility among Combinatorial Problems," *Complexity of Computer Computations*, 85–104, Miller, R. E., and Thatcher, J. W., (eds.), Plenum Press, New York.

Karp, R., (1975). "On the Complexity of Combinatorial Problems," *Net-*

works, **5**, 45–68.

Lawler, E., (1979). "Fast Approximation Algorithms for Knapsack Problems" *Math. Opns. Res.* **4**, 339–356.

Lawler, E., (1976). *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.

Sahni, S. (1976). "Algorithms for scheduling independent tasks," *J. Assoc. Comput. Mach.*, **23**, 116–127.

Sahni, S., and Horowitz, E., (1978) "Combinatorial Problems: Reducibility and Approximation," *Opns. Res.* **26**, 718–759.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 81-9	2. GOVT ACCESSION NO. AD-A105881	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Exact and Approximation Algorithms for a Scheduling Problem		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Gregory Dobson		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		12. REPORT DATE July 1981 13. NUMBER OF PAGES 37
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

COMBINATORICS
POLYNOMIAL
ALGORITHM EXACT
ALGORITHM APPROXIMATE
ASSIGNMENT OF JUDGES
BIN PACKING

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 81-9: "Exact and Approximation Algorithms for a Scheduling Problem," by Gregory Dobson

This paper discusses problems that arose in calendaring cases for an appellate court. The first problem is to distribute cases among panels of judges so as to equalize work loads. We give a worst case analysis of a heuristic for this NP-complete problem. For a given distribution denote by z the heaviest work load. We wish to minimize z. The ratio of the heuristic value $\bar{z}$ to that of the true optimum z^* is shown to be $\bar{z}/z^* \leq (k+3)/(k+2)$ where all the case weights are in $[0, (1/k)z^*]$, generalizing a result of Graham on multiprocessor scheduling. Under a restrictive assumption on the case weights, some generalizations of this scheduling problem are solved. Characterizations for feasible calendars and polynomial algorithms for finding these feasible solutions are given. Algorithms are given for choosing an optimal subset of the backlogged cases that can be calendared.

END

DATE
FILMED

11-81

DTIC